IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of:	)	Confirmation No.: 8871
	)	
Ashish Thusoo	)	Examiner: Cory C. Bell
	)	
Serial No.: 10/662,095	)	Group Art Unit No.: 2164
	)	
Filed on: September 12, 2003	)	
	)	

For: AGGREGATE FUNCTIONS IN DML RETURNING CLAUSE

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL APPEAL BRIEF

Sir:

This Supplemental Appeal Brief is submitted subsequent to the Appeal Brief filed on March 30, 2007, which was submitted in support of the Notice of Appeal filed on March 22, 2007.

CERTIFICATE OF TRANSMISSION VIA EFS-WEB

Pursuant to 37 C.F.R. 1.8(a)(1)(ii), I hereby certify that this correspondence is being transmitted to the United States Patent & Trademark Office via the Office electronic filing system in accordance with 37 C.F.R. §§1.6(1)(4) and 1.8(a)(1)(i)(C) on the date indicated below and before 9:00 PM PST.

Submission date: August 21, 2007 by /DanielDLedesma#57181/

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

Oracle International Corporation is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-20 and 22-25 are pending in this application and were finally rejected in the Final Office Action mailed on December 22, 2006. Claim 21 was canceled during prosecution.

Claims 1-20 and 22-25 are the subject of this appeal.

IV. STATUS OF AMENDMENTS

In the Office Action mailed on December 22, 2006, Claims 15-20 and 22-25 were rejected under 35 U.S.C. § 101 as being allegedly non-tangible as it can be embodied in a carrier wave. Accordingly, as agreed to by the Examiner, an amendment was filed on March 28, 2007. In the amendment, Claim 15 was amended to recite "wherein the computer-readable medium is one of a volatile medium or non-volatile medium" in order to overcome the 35 U.S.C. § 101 rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present application contains independent Claims 1 and 15. Claim 1 is a method claim. Claim 15 is a computer-readable medium claim and is a counterpart of method Claim 1.

Claims 1 and 15 are generally directed to an approach for receiving a database statement that (1) specifies a data manipulation language (DML) operation that modifies data and (2)

contains a clause that specifies an aggregate operation to be performed on multiple values associated with the data.

A. BACKGROUND

It is often desirable for a user performing a DML operation to know what values were involved in the operation and to perform subsequent operations on those values. For example, a user may want to know the old values that were deleted in a delete operation, the old values that were overwritten in an update operation, and/or new values that were written in an update operation. For the purpose of explanation, the set of values involved in a prior operation, which are to be used in a subsequent operation, are referred to herein as "values of interest" (e.g., the "values-of-interest" can be which ever of the old values and the new values are of interest for performing aggregation operations).

In database systems that support SQL, a "returning" clause may be used to obtain a value-of-interest. For example, assume that a user wants to increment a value in a column, C1, of particular row (where key=5), and have the database server return the new value that was written during the update operation. The following code segment could be used to perform the desired operation:

```
update T
        set C1 = C1 + 1
        where key = 5
returning C1
```

Unfortunately, the conventional implementation of the returning clause only returns a single value-of-interest. Therefore, programmers must resort to more complex techniques in situations where (1) a first operation changes many values, thereby creating many values-of-interest, and (2) an aggregate operation is to be performed on the values-of-interest associated with the first operation. One such technique is referred to herein as the iterative technique.

According to the iterative technique, the first operation (which changes many values) is performed by iteratively executing code that only changes a single value. During each iteration, the returning clause is used to return the value-of-interest associated with that iteration. For example, assume that a user wants to increment several values in C1, and to keep track of all of the new, post-update values that are changed during the update operation. Such an operation could be performed using the following code segment.

```
For All i=1....N
        update T
                set C1 = C1 + 1
                where key = :input[i]
returning C1 in :output [i];
```

This code segment updates values in the column C1 of table T by adding a 1 to the numbers stored in column C1 of various rows of table T. The statement "where key=:input [i]" above is an instruction to use the array input[] to determine which rows are updated.

For example, assume that input[] stores the following values:

```
input[1]=1
input[2]=3
input[3]=7
```

and that table T initially has the following rows:

KEY	C1
1	23
2	45
3	2
4	108
5	95
6	9
7	10

During the first iteration, the "where" clause will be "where key=1", since i=1 and input[1]=1. Thus, during the first iteration, the value 23 in the first row of table T is updated to 24, and the value 24 is stored in output[1].

During the second iteration, since i=2 and input[2]=3, the value 2 is updated to 3 in the third row of table T of C1, and the value 3 is stored in output[2]. During the third iteration, since i=3 and input[3]=7, the seventh element of C1, 10, is updated to 11 and the value 11 is stored in output[3]. Thus, after finishing first three iterations, output[] has the values [24,3,11], and the updated table T has the following values:

KEY	C1
1	24
2	45
3	3
4	108
5	95
6	9
7	11

Once all of the values-of-interest are captured in the output array, aggregate values may be computed based on the captured values-of-interest. For example, the sum of the captured values-of-interest may be computed by executing the code segment:

```
s=0;
For i =1...N
        s=s+:output[i];
```

This code segment initializes the variable s to have the value 0, with the statement s=0. Then, in the for-loop for each element specified by the output array, output[i], the value of the specified element is added to the variable s. The end result is that s will contain a value that represents the sum of all values-of-interest that were captured in output[].

It should be noted that in the present example, the user was only interested in the sum of the updated values, not in the individual updated values themselves. However, for the sum of the updated values to be calculated, all of the values-of-interest are returned. Performing aggregate functions using the iterative approach not only returns unnecessary data from the server to the client but may also require an evaluation of the aggregation by an application on the client side. Thus, conventional methods of performing aggregate functions on values-of-interest are inefficient. Therefore, it is desirable to provide techniques for performing aggregate functions on values-of-interest that do not involve the programming complexity and wasteful data movement as experienced using conventional techniques.

B. CLAIMS 1 AND 15

Claims 1 and 15 feature receiving a database statement that (a) specifies a DML operation that modifies data (Application, paragraph [0003]) and (b) contains a clause that specifies an aggregate operation to be performed on a plurality of values associated with the data. (Application, paragraphs [0006] and [0025]). In response to receiving the database statement, the DML operation and the aggregate function are performed and a result of the aggregate operation is returned. (Application, paragraphs [0025] and [0027]).

For example, using the syntax of a new returning clause, the aggregation operation discussed above for performing a summation of elements of column C1 may be replaced with:

```
Update T
        Set C1=C1+1
        Returning sum(C1) into s;
```

In the code segment example, above, because the aggregation is performed before sending the data back to the client, the amount of data to be returned is significantly less than

the amount of data that is returned in the iterative approach described above. (Application, paragraphs [0025]-[0026]).

Independent Claim 15 is a computer-readable medium counterpart of method Claim 1, and includes limitations analogous to the limitations of Claim 1. Thus, the elements of Claim 15 are disclosed in at least the same sections of the specification as those cited above in connection with Claim 1. In addition, the elements of Claim 15 are supported by the hardware and computer-readable media description provided on pages 15-16, paragraphs [0035]-[0037] and elements 304 and 310 of FIG. 3 of the Application.

No dependent claim is argued separately.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-7, 10, and 15-20 have been rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over PL/SQL User's Guide and Reference Release 2(9.2) ("reference [A]") in view of Applicant admitted prior art ("AAPA"). Reference [A] includes: (a) pages 51-55 of Chapter 5, entitled "PL/SQL Collections and Records"; and (b) pages 1-13 of Chapter 12, entitled "Tuning PL/SQL Applications".

Claims 8-9, 11, 13, 22, and 24 have been rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over reference [A] in view of Oracle Corporation, "Oracle9i SQL Reference, Release 2(9.2) ("reference [B]").

Claims 12 and 23 have been rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over reference [A] in view of reference [B], and further in view of U.S. Patent No. 6,567,803 issued to Ramasey et al. ("*Ramasey*").

VII. ARGUMENTS

It is respectfully submitted that the Examiner has erred in rejecting Claims 1-20 and 22-25 under 35 U.S.C. §103(a).

A. CLAIMS 1 AND 15

Claims 1 and 15 feature:

"receiving a database statement that

specifies a data manipulation language (DML) operation that modifies data in one or more columns in a database, and

contains a **clause that specifies an aggregate operation** to be performed on a plurality of values associated with the data, wherein each of the plurality of values are from a separate row; and

in response to receiving the database statement,

performing the DML operation on the one or more columns in the database,

performing the aggregate operation on the plurality of values, and

returning as a result of the database statement a result of the aggregate operation." (emphasis added)

The Office Action (mailed on December 22, 2006) admits that [A] fails to "disclose the use of an aggregate function in a returning clause. The Office Action then asserts that "it would have been obvious to one of ordinary skill in the art to include aggregate functions as presently amended. This is incorrect.

As support for this assertion, the Office Action states that:

> First, the RETURNING clause is intended to eliminate the need for a select clause, [A] page 9 '*This eliminates the need to SELECT the row after an insert or update, or before a delete.*' Thus to truly eliminate the need for the SELECT clause it would need to integrate all the features of the SELECT clause, i.e. the ability to perform aggregate functions as claimed. Which are used to, as stated on pages 9 and 10 of reference [A], '*As a result, fewer network round trips, less server CPU time, fewer cursors, and less server memory are required.*' Thus, it would have been obvious to one of ordinary skill in the art to provide these features in the return clause to provide the advantage of full eliminating the need

> for the select after an insert or update or before a delete. Reference [A] also shows the need to aggregate data further as it states '*Now do computations involving name and new_sal.*' (Office Action, pages 3-4; italics in original).

It is respectfully submitted that the Office Action is incorrect on at least several counts.

MPEP 2143 states that, in order to establish a prima facie case of obviousness, three criteria must be met. "First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations."

As will be shown hereinafter, the combination of the motivation and reference [A] fails to teach or suggest all the claim limitations. Furthermore, even the cited motivation is taught away by a statement in reference [A].

1. The Cited Reference and Motivation Fail to Teach or Suggest All the Features of Claims 1 and 15

First, reference [A] provides a benefit (referred to hereinafter as the "cited motivation") of including a RETURNING clause for returning column values from an affected row after or before certain database statements. The cited motivation is as follows: "As a result, fewer network round trips, less server CPU time, fewer cursors, and less server memory are required" (pages 9-10 of Chapter 12). The Office Action cites this languages and then asserts, "Thus it would have been obvious to one of ordinary skill in the art to provide these feature[s] in the [RETURNING] clause to provide the advantage of full[y] eliminating the need for the [SELECT statement] after an insert or update or before a delete" (page 4).

However, based on the cited motivation, *at most* the only obvious modification to the reference [A] would be to allow the RETURNING clause to return multiple values from

different rows. It would not have been obvious to allow the RETURNING clause to *specify* an aggregate operation to be performed on the multiple values from the different rows. Indeed, reference [A] fails to teach or suggest a RETURNING clause specifying *any* operation to be performed on values associated with modified data. Moreover, the Office Action **fails to provide any such suggestion or motivation**. Therefore, the combination of reference [A] and the cited motivation fail to teach or suggest *all* the claim limitations of Claims 1 and 15.

2. *The Cited Reference Teaches Away From the Suggestion or Motivation*

It would not have been obvious to even modify reference [A] to disclose a RETURNING clause that returns column values from *multiple* affected rows, much less to modify reference [A] to disclose a RETURNING clause that specifies an aggregate operation. MPEP 2141.02(VI) states: "A prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention. *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983)" (emphasis in original).

Reference [A] specifically states, "You can use [the RETURNING] clause ***only*** when operating on exactly **one row**." (page 53 of Chapter 5, entitled "PL/SQL Collections and Records"). Therefore, reference [A] **teaches away** from using a RETURNING clause when values from multiple rows are modified. MPEP 2141.02(VI), MPEP 2143.01(I), and MPEP 2144.05(III) state that the teaching away statement(s) should at least discourage one or ordinary skill in the art from modifying the cited reference(s). It is respectfully submitted that the teaching away statement in reference [A] rises to the level of discouraging one of ordinary skill in the art to modify reference [A] as alleged. Indeed, after reading the teaching away statement in reference [A], one of ordinary skill in the art would attempt to use a different method for

performing an aggregate operation on values from separate rows, such as the iterative approach described in the Background section above (supra V(A)).

Even if one of ordinary skill in the art were to ignore the teaching away statement in reference [A], that person would not be motivated to use an aggregate operation on the multiple values associated with modified data. Indeed, as stated previously, one of ordinary skill in the art would *at most* be motivated to modify the RETURNING clause to return multiple modified values from different rows, but would not be motivated to modify the RETURNING clause to specify an operation, much less an aggregate operation.

3. The Cited Reference Fails to Teach or Suggest Integrating All Features of the SELECT Statement

The Office Action cites reference [A] for teaching: "[Including a RETURNING clause in INSERT, UPDATE, and DELETE statements] eliminates the need to **SELECT *the row*** after an insert or update, or before a delete" (page 9 of Chapter 12; emphasis added). Thus, the RETURNING clause is used only in this specific instance when a single row is modified. The Office Action then alleges that "to truly eliminate the need for the SELECT clause [the RETURNING clause] would need to integrate all the features of the SELECT clause, i.e. the ability to perform aggregate functions as claimed" (page 4). (It is respectfully noted that "SELECT clause" should read "SELECT statement").

It does not necessarily follow that the RETURNING clause would integrate all the features of the SELECT statement simply because reference [A] states that the RETURNING clause eliminates the need to SELECT a *single row* in a *specific instance*. Such an inference requires multiple deductive steps that are missing from the Office Action and the cited references.

4. The Cited Reference Fails to Show "the Need to Aggregate Data Further"

The Office Action further alleges that "Reference [A] also shows the need to aggregate data further as it states '*Now do computations involving name and new_sal*'" (page 4; italics in original). However, as recited in Claim 1, and made clear through responses to the previous Office Actions, no aggregation of 'name' and 'new_sal' can be performed because 'name' and 'new_sal' represent individual values from the **same** row. Even if the Office Action is suggesting that the variables 'name' and 'new_sal' are somehow combined (a dictionary definition of aggregate), reference [A] lacks any teaching or suggestion of such a combination.

5. Summary

Based on the foregoing, all the features of Claims 1 and 15 are not taught or suggested by reference [A]. It would not have been obvious to one of ordinary skill in the art at the time of the invention to modify reference [A] to include a (e.g., RETURNING) clause, in a database statement, that specifies an aggregate operation because reference [A] teaches that a RETURNING clause can only be used when operating on exactly one row. Even if this teaching away language would have been ignored by one of ordinary skill in the art, the combination of reference [A] and the cited motivation would at most only suggest a RETURNING clause that returned multiple values from different rows, and **not** a RETURNING clause that specified an aggregate operation on those multiple values.

B. CLAIMS 2-14, 16-20, AND 22-25

Claims 2-14 are dependent upon Claim 1, and Claims 16-20 and 22-25 are dependent upon Claim 15. Thus, each of Claims 2-14, 16-20, and 22-25 include each and every feature of the corresponding independent claims. Therefore, the Applicant respectfully submits that each of Claims 2-14, 16-20, and 22-25 is therefore allowable for the reasons given above for Claims

1 and 15. In addition, each of Claims 2-14, 16-20, and 22-25 introduces one or more additional limitations that independently render it patentable. A full discussion of each dependent claim is not included herein at this time based on the fundamental differences already identified herein.

C. CONCLUSION AND PRAYER FOR RELIEF

Based on the foregoing, it is respectfully submitted that the rejection of Claims 1-20 and 22-25 under 35 U.S.C. § 103(a) as being unpatentable over the cited art lacks the requisite factual and legal bases. Appellants therefore respectfully request that the Honorable Board reverse the rejection of Claims 1-20 and 22-25 under 35 U.S.C. § 103(a).

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

/DanielDLedesma#57181/
Daniel D. Ledesma
Reg. No. 57,181

Date: August 21, 2007
2055 Gateway Place, Suite 550
San Jose, CA 95110-1089
Telephone: (408) 414-1229
Facsimile: (408) 414-1076

VIII. CLAIMS APPENDIX

1. (previously presented) A method comprising:

receiving a database statement that

specifies a data manipulation language (DML) operation that modifies data in one or more columns in a database, and

contains a clause that specifies an aggregate operation to be performed on a plurality of values associated with the data, wherein each of the plurality of values are from a separate row; and

in response to receiving the database statement,

performing the DML operation on the one or more columns in the database,

performing the aggregate operation on the plurality of values, and

returning as a result of the database statement a result of the aggregate operation.

2. (previously presented) The method of claim 1, wherein the performing of the aggregate operation is performed while performing the DML operation.

3. (previously presented) The method of claim 1, wherein the modified data includes values of the data before the DML operation.

4. (previously presented) The method of claim 1, wherein the modified data includes values of the data after the DML operation.

5. (previously presented) The method of claim 1, wherein the DML operation is an update of the data.

6. (previously presented) The method of claim 1, wherein the DML operation is a deletion of the data.

7. (original) The method of claim 1, wherein the receiving is performed by an SQL engine.

8. (original) The method of claim 1, wherein results of the aggregate operation are passed from an SQL engine to a server side stub without passing the data in its entirety.

9. (original) The method of claim 8, wherein results of the aggregate operation are passed from an SQL engine to a client interface without passing the data in its entirety.

10. (original) The method of claim 1, where the database statement is sent from a client interface.

11. (original) The method of claim 1, wherein the database statement contains multiple aggregate operations.

12. (previously presented) The method of claim 11, wherein performing the aggregate operation includes:

parsing the database statement;

establishing a list of operator trees, each operator tree corresponding to a different aggregate function; and

establishing an aggregate function list including structures pointing to work spaces for performing the aggregate functions.

13. (original) The method of claim 1, wherein:

the receiving of the database statement is performed via a call interface;

the performing of the aggregate operation generates an aggregate value; and

the method further includes passing the aggregate value through the call interface without passing the plurality of values.

14. (previously presented) The method of claim 1, wherein:

the database statement further contains a return clause that indicates old values associated with the data; and

performing the aggregate operation on the plurality of values includes performing the aggregate operation on the old values.

15. (previously presented) A computer-readable medium storing a set of instructions, wherein the computer-readable medium is one of a volatile medium or a non-volatile medium, wherein the set of instructions, when executed by one or more processors, causes the one or more processors to perform a method including at least:

receiving a database statement that

specifies a data manipulation language (DML) operation that modifies data in one or more columns in a database, and

contains a clause that specifies an aggregate operation to be performed on a plurality of values associated with the data, wherein each of the plurality of values are from a separate row; and

in response to receiving the database statement,

performing the DML operation on the one or more columns in the database,

performing the aggregate operation on the plurality of values, and

returning as a result of the database statement a result of the aggregate operation.

16. (previously presented) The computer-readable medium of claim 15, wherein the performing of the aggregate operation is performed while performing the DML operation.

17. (previously presented) The computer-readable medium of claim 15, wherein the modified data includes values of the data before the DML operation.

18. (previously presented) The computer-readable medium of claim 15, wherein the modified data includes values of the data after the DML operation.

19. (previously presented) The computer-readable medium of claim 15, wherein the DML operation is an update of the data.

20. (previously presented) The computer-readable medium of claim 15, wherein the DML operation is a deletion of the data.

21. (cancelled).

22. (previously presented) The computer-readable medium of claim 15, wherein the database statement contains multiple aggregate operations.

23. (previously presented) The computer-readable medium of claim 22, wherein performing the aggregate operation includes:

parsing the database statement;

establishing a list of operator trees, each operator tree corresponding to a different aggregate function; and

establishing an aggregate function list including structures pointing to work spaces for performing the aggregate functions.

24. (previously presented) The computer-readable medium of claim 15, wherein:

the receiving of the database statement is performed via a call interface;

the performing of the aggregate operation generates an aggregate value; and

the set of instructions includes further instructions which, when executed by the one or more processors, further cause the one or more processors to perform passing the aggregate value through the call interface without passing the plurality of values.

25. (previously presented) The computer-readable medium of claim 15, wherein:

the database statement further contains a return clause that indicates old values associated with the data; and

performing the aggregate operation on the plurality of values includes performing the aggregate operation on the old values.

IX. EVIDENCE APPENDIX PAGE

None.

X. RELATED PROCEEDINGS APPENDIX PAGE

None.